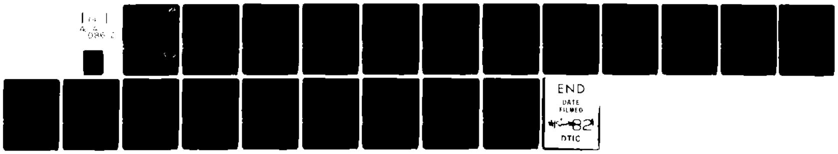
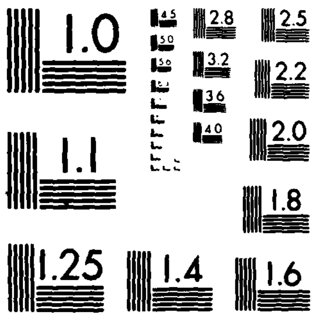
AD-A108 612   DELAWARE UNIV NEWARK DEPT OF COMPUTER AND INFORMATI--ETC F/G 5/2
PREREQUISITES TO DERIVING FORMAL SPECIFICATIONS FROM NATURAL LA--ETC(U)
OCT 81 R M WEISCHEDEL   AFOSR-80-0190
UNCLASSIFIED   AFOSR-TR-81-0798   NL




1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

Prerequisites to Deriving Formal Specifications from Natural Language Requirements
Final Report*

by

Ralph M. Weischedel

October, 1981

*Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant no. AFOSR-80-0190. Th United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation herein.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 81-0798	2. GOVT ACCESSION NO. AD-A108 612	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PREREQUISITES TO DERIVING FORMAL SPECIFICATIONS FROM NATURAL LANGUAGE REQUIREMENTS		5. TYPE OF REPORT & PERIOD COVERED FINAL 7/1/80-8/31/81
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ralph M. Weischedel		8. CONTRACT OR GRANT NUMBER(s) ✓ AFOSR-80-0190
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer & Information Sciences, University of Delaware, Newark, DE 19711		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332		12. REPORT DATE October, 1981
		13. NUMBER OF PAGES 19
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

formal specifications, English specifications, modules, software design, natural language processing, software engineering

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Since English specifications and formal specifications of modules are complementary and since formal specifications require so much effort to write, our work is investigating application of artificial intelligence techniques to aid in the software specification process.

The effort for this year involved four areas of work. The first is comparing English descriptions with formal specifications of the same software module. This work is now complete; however, some of the examples will continue to serve as a guide to the software tool being constructed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED:

The second area is suggesting modifications to formal specification languages, which would make them more understandable. In particular, we have been suggesting alternatives to logical quantifiers. This will continue next year.

The third and fourth areas deal with the syntactic and semantic components of an experimental software tool. Its purpose is to test our solutions to a handful of problems in transforming English descriptions to formal specifications under significant user assistance. This work is in progress this year, but will not be complete until the end of another year's effort.

In dealing with syntactic ambiguity, one of the solutions that many have speculated about is the use of a semantic component to reject anomolous parses; we intend to test its effectiveness using the RUS grammar. When ambiguity is not resolved, questions must be presented to the user for his/her selection of the intended interpretation. Partial heuristics for this are part of the results of this year's effort. They will be implemented and tested during the next year. In addition, modification of the RUS parser and dictionary for the domain of software specification will continue during the next year.

Another problem is reflecting the conceptual view of a person in the semantic component. This must then be translated into the more mathematical view of present specification languages. On this problem, we have focused on designing this conceptual view; in the next year we will implement the translation from it to the mathematical level. A second semantic problem is anaphora resolution: the heuristics of Sidner (1979) are being modified for dealing with texts of specifications rather than dialogues (the class of discourse for which they were originally developed). This work will also continue next year.

0. Organization of the Report

This final report of grant AFOSR-80-0190 is divided into five sections. Section one gives a brief summary of the motivation of the effort. Section two reports our progress this year and briefly proposes the work for the coming year. Section three summarizes the report. The references are collected in section four; a list of publications and presentations during this year's effort appears in section five. Related work is cited throughout section two; however, a complete discussion of related work appears in Weischedel (1980) and in the proposal that resulted in grant AFOSR-80-0190.

1. Motivation of the Work

The technique of formal module specifications seems to offer much toward alleviating many problems of large software systems (those systems requiring at least 25 programmers for development and at least 30,000 lines of source code). The high cost of software maintenance, the predominance of design errors, the difficulty in modifying software, and the difficulty and cost of diagnosing and correcting design errors are some of the problems addressed by formal specifications based on the information-hiding principle. Yet, the creating of formal specifications is very difficult, requiring much upfront effort. For instance, Parnas (1976, p. 7) states, "Experience has shown that the effort involved in writing the set of specifications can be greater than the effort it would take to write one complete program." It is also generally agreed that formal specifications are difficult to understand.

Our work, which has been funded by AFOSR under contract number F49620-79-0131 and grant number AFOSR-80-0190, has had both long-term and short-term goals. The long-term objectives include a preliminary, fundamental study of the problems involved in understanding software system requirements written in English and transforming them into formal specifications of a software module. The research is now concentrating on a few of the basic problems our work has identified and has begun to develop a small prototype system to test our proposed solutions.

One aspect of our study has been a comparison of English descriptions and formal specifications of the same software modules. This has provided a basis for achieving our short-term goals: suggestions for better documentation and insights into the reasons that software specifications are difficult to understand.

The motivation of the work is covered at greater length in the proposal for the grant and the contract mentioned above.

2. Progress and Future Work

The proposal for this present year's effort included four areas; two in short-term-goals and two in long-term goals.

2.1 Short-term Goals

The following two sections deal with the short-term goals: identifying why formal specifications are so difficult to understand, suggesting better documentation techniques, and formulating an alternative to one difficult mechanism in formal specifications.

The formal specification of a module must be understandable if it is to achieve its purpose, for it acts as a contract between designers and programming team, stating exactly what the programming team's product must do (Parnas, 1977). Unless they are understandable, 1) programmers will not know what the module they are to implement is to do nor how to use other modules, and 2) designers will not be able to detect design errors nor easily confirm that their design satisfies user requirements. Also, if one is to use a reference library of formal specifications, they must be understandable, for if the designer cannot understand the alternative specifications, how can an intelligent choice be made among the alternatives? If formal specifications of module interfaces are to become widely used, they must be understandable.

2.1.1 Comparison of English descriptions and formal specifications

In our comparison of specifications in English versus those in formal languages, we have used examples from Horowitz and Sahni (1976); all of the examples have been small, typical data structures. These are representative of the class of problems that the system described in 2.2 should be able to handle in the next few years. These examples complement our earlier study of portions of KSOS (Ford Aerospace, 1978), since KSOS, the kernel of a secure operating system, is one of the largest systems ever formally specified.

We have discovered a number of causes for the difficulty in understanding formal specifications. For instance, one is the difference between the conceptual view expressed in English and the mathematical view required in formal specifications. Rather than viewing a stack as a mathematical entity, such as a finite sequence, the description in Horowitz and Sahni (1976) takes a more spatial view by stating (page 77), "A stack is an ordered list in which all insertions and deletions are made at one end, called the top. Given a stack S=(a1, ... , an) then we say that a1 is the bottommost element and element ai is on top of element ai-1, $1<i\leq n$."

In addition, we have several recommendations regarding better documentation of formal specifications. These, along with our results on understandability are reported elsewhere (Weischedel, 1981a and 1981b). This work is complete; however, the examples from Horowitz and Sahni (1976) will continue to serve as a basis for development of the software tool (see section 2.2).

2.1.2 Quantification in formal specifications

The proposal for this year's work *identified quantification* in formal logic both as a source of difficulty in understanding formal specifications and as a mechanism for which an alternative could be found. Psychological evidence (Johnson-Laird, 1980; Thomas, 1976; Wason and Johnson-Laird, 1972) supports the idea that people naturally understand things by means of models rather than by abstract inferences. These models consist of sets of objects and the relationships between them. When these relationships are modelled so as to change through time, the models serve as simulations. People can then make predictions by observing what happens in their models. This mode of reasoning is more direct and innate than the deductive mode of classical logic, the mode on which most program specification languages are based.

Models are uncomplicated representations when compared to our sophisticated formalisms because in one sense they have no quantifiers, all their component facts are atomic relations between individual objects, and the only logical connective is conjunction. Models may be thought to convey quantified information however; to the extent that the objects in them are representative cases, models are generalizations. We are developing models that are an alternative to quantification, yet suitable for specification.

Since generalizations are often stated in English, its grammar provides for quantification; we are examining some of those properties as possible modifications to specification languages. The objects talked about are often indefinite, arbitrary or generic; such objects can be conveniently represented by variables. It is often understood that the identity of some indefinite objects cannot be determined independent of other objects; such an object can be represented by Skolem functions, which explicitly state this dependency. For instance, file names are such objects in "Every file has a file name". Object references (noun phrases) are usually dominated by relationships (verbs) instead of the opposite, which is the case in present formal specifications. Object references also contain any restrictions or presuppositions that may be intended. We suggest using expressions like

(any X such that X IS-OPEN-FILE) IS-USER-FILE

rather than the more traditional form:

for all X (if IS-OPEN-FILE(X) then IS-USER-FILE(X)).

The alternative we are developing has some problems. The main one is quantifier scope. There are times when a variable, which is always universally quantified, must be limited in its scope, as when a general statement is negated or embedded in another statement. English has some ad hoc limiting devices which we may adopt. To illustrate just one device, note that English has several words to indicate universal quantification, including 'every' and 'any'. These words differ, however, in that 'every' seems to take minimal scope, while 'any' seems to take maximal scope. Compare the sentences

a. If any file is open, the user must close it.

b. If every file is open, the user must close it.

The 'it' in (b) cannot refer to the files, because the scope of 'every' is restricted to only the condition stated in (b). We are studying these ad hoc devices to see how they may fit into a formal language and to see what their limitations are.

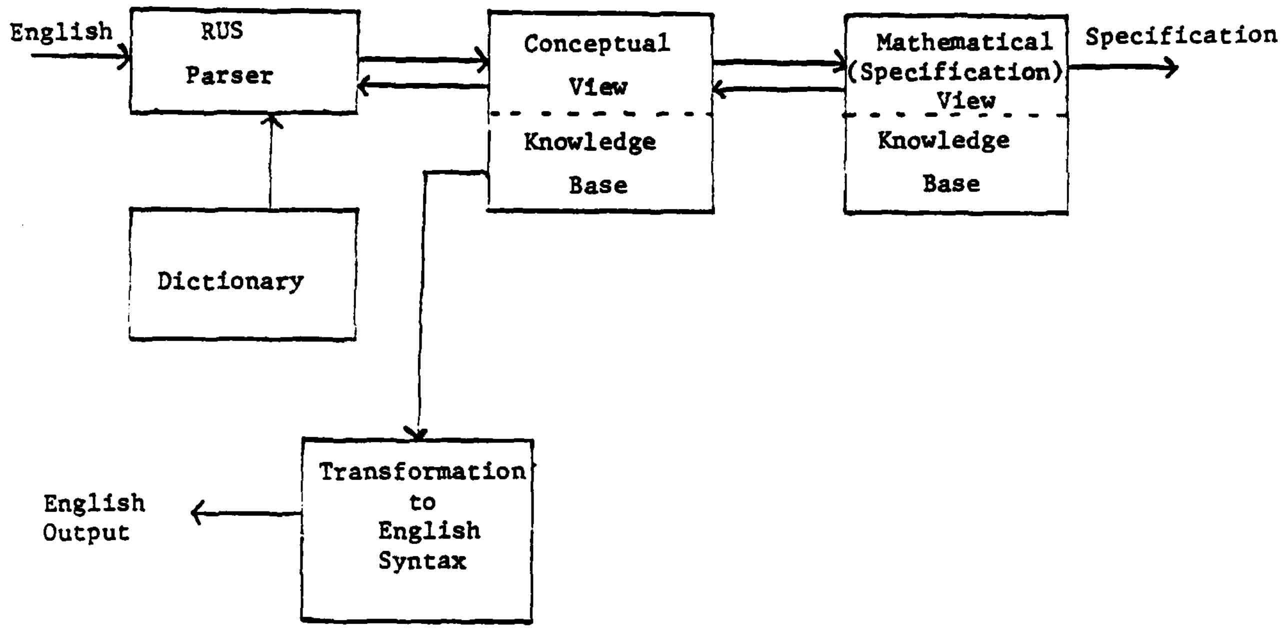
A very useful modification we are also investigating would greatly increase the readability of formal specifications. It is fairly easy to suggest a syntax which appears like a very restricted subset of English. This makes the language much more readable. An example is JARGON (Woods, 1979), which has been suggested as a language for designing semantic networks in artificial intelligence. The same approach should apply to formal specification languages.

Though we had originally planned on terminating this work on alternatives to quantification this year, our progress is leading to more interesting possibilities than we originally projected. The potential impact on making formal specifications more understandable and more easy to write warrants continued effort.

2.2 Long-term Goals

The two sections that follow deal with the long-term goals. In particular, Weischedel (1980) *identifies the form* a tool might have, which, under significant user direction, would derive formal specifications from English requirements. Though that work could not project when, if ever, such a tool would exist for specifying large software systems, it did identify a few fundamental problems that are prerequisite to such a tool existing. We are developing a very small prototype tool to test our hypothesized solutions to those problems.

The structure we have chosen for the prototype tool is given in Figure 1. All arrows represent data flow.



SYSTEM STRUCTURE

Figure 1

Given an English input, the parser recognizes the syntactic structure of it. Using a lexicon (a highly formalized dictionary), the parser generates a semantic representation for the input. Since the syntax of English is highly ambiguous, the parser interacts with the knowledge base in the "conceptual view" as a means of using factual knowledge to reduce the number of semantically senseless parses tried by the parser. The parser and dictionary constitute the syntactic component of the system.

In section 2.1.1 we argued that the level at which people conceive of a module is often quite different than the mathematical level that specification languages presume. This is reflected in the system by having two distinct views: a conceptual one using terms close to that of the human specifier and a mathematical one close to that of specification languages. At both levels, basic facts and rules of inference will have to be stored in order to reason about the semantic representation that is being constructed from the user's description of the module. These facts and rules of inference are called a knowledge base.

The transformation from the conceptual view to the mathematical one will probably use techniques similar to Barstow (1977). It is the transformation between the two views that will detect some of the missing information in the user's specification. Questions eliciting the missing information must be generated using the coneptual view since that is the level using terms closest to the user's. A simple state-of-the-art output component can transform the notation of the conceptual level to English for asking questions and explaining the system's interpretation of the input. The two views constitute the semantic component of the system.

2.2.1 Syntax

For the syntactic component of the prototype software tool we are developing, we are using RUS (Bobrow, 1978), a grammar of English which calls a semantic component while parsing in order to cut down on semantically anomolous parses. Since it is a general grammar of English, some extensions must be made to understand expressions that are natural in software specification though not in everyday language. The simple mathematical notation in the quote in section 2.1.1 is an example. Additionally, a dictionary for terms commonly used in software specification is under construction for use with the grammar. For instance, the use of 'say' in "We say that a stack is ..." is rather stylized in software specifications as a way of defining a term. Work on the dictionary and the parser will continue next year.

In addition to modifying the parser and developing a dictionary, this year's work included postulating heuristics for generating clarifying questions when the system cannot resolve ambiguity. Weischedel (1980) identified several difficult classes of ambiguity that occur in software specifications. One is prepositional phrase attachment. The paradigmatic example of this in the literature is "I saw the man in the park with a telescope." There are a significant number of cases where a system could not resolve such ambiguity, but must instead ask the user which interpretation is intended. Clearly, two things are needed: a means of detecting that two interpretations are present and a technique of <u>precisely</u> stating the alternative interpretations. (If the statements given to the user for his/her selection are ambiguous, this only compounds the problem.)

We have found the cleft construction in English plus fronted adverbials to be a promising way of distinguishing the alternatives. For the example above, the various interpretations are

1. It was the man in the park having a telescope that I saw.
2. Using a telescope it was the man in the park that I saw.
3. It was in the park that I saw the man having a telescope.
4. It was in the park and using a telescope that I saw the man.

We will continue to study the potential of this technique by implementing the heuristics next year.

2.2.2 Semantics

The proposal that funded this year's effort identified the need to have a knowledge representation intermediate in level between the informal English description and the formal (and rather mathematical) specification. Additional support for this is the observation in section 2.1.1 above that the human's conceptual view of the module being defined can be quite different than the formal specification. Mirroring the human's conceptual view as one semantic level will provide the right level for phrasing questions put to the user to fill in missing detail and for phrasing the system's understanding of the English input.

Much of our effort in semantics has been designing the knowledge representation for those two levels. Both levels will be expressed in first-order logic or its variations; however, the terms (i.e. predicates, constants, functions) will be quite different at each level.

As an example, consider the following definition of an insertion operation for ordered lists, as given on page 42 of Horowitz and Sahni (1976):

"insert a new element at position i, 1<i<n+1 causing elements numbered i,i+1,...,n to become numbered i+1,i+2,...,n+1".

A formal logic expression capturing its semantics using terms close to the person's view follows in a:

```
a:  (FORALL y6:  LAMBDA(y7:element).new(y7)).
     cause[insert-at(UNKNOWN,y6, iz6:(position(z6) &
                                      z6=i & 1<=i
                                      & i<=n+1)),
           (FORALL w IN {1 ... n}).
            become (i w1:(element(w1) & numbered(w1,w)),
                  numbered(w1,w+1))]
```

Though the details are unimportant, it should be noted that it uses terms such as 'position' that are very close to English words (and the associated concepts). On the other hand, at the mathematical level, formula b is appropriate and is semantically very close to a formal specification language such as SPECIAL (Roubine and Robinson, 1976).

```
b:  (LAMBDA  x: ordered-list(x),p: timeinterval(p),e,
             i: 1<=i & i<= length(x, begin(p))+1)
     rep(x, end(p))=i f: finite-seq(x) &
                         (FORALL j: 1<=j & j<= length(x,begin(p))+1)
                         [(j<i -> f(j)=rep(x,begin(p))(j)) &
                           f(i)=e &
                          (j>i -> f(j)=rep(x,begin(p))(j-1))]
```

We will continue the design of these two levels of representation during the next year. In addition, the mapping from the conceptual view to the mathematical view will be implemented.

In addition to the semantic analysis discussed above, a separate problem in semantics is being studied. Determining what a pronoun or noun phrase refers to has been a topic of much interest recently in artificial intelligence; it is called definite anaphora resolution. Sidner (1979) presents a heuristic for detecting clues in dialogue for the focus of attention, and explains how that can be used for determining reference. The consultant on this grant has been modifying this heuristic for texts of English specifications rather than dialogue; see Joshi and Weinstein (1981). In doing this, he has found that Sidner's notion of focus is inadequate. Rather, two objects seem to serve as centers of attention. A backward center serves as a locus in a single sentence and corresponds to previous objects in the text. A forward center serves as a focus for later references and introduces new objects. In the sentence "Processes must execute in a single fork", "processes" serves as the backward center, and "a single fork" is the forward center. Furthermore, he is postulating heuristics for detecting centers based on the structure of a sentence recognized by a parser. A practical application we expect from this is an additional criterion for determining whether an English specification is well-organized and well-written based on its anaphoric references. This study will continue next year using English specifications from PSOS (a provably secure operating system) (Neuman, et al., 1977) and from Horowitz and Sahni (1976).

3. Conclusions

The effort for this year involved four areas of work. The first is comparing English descriptions with formal specifications of the same software module. This work is now complete; however, some of the examples will continue to serve as a guide to the software tool being constructed.

The second area is suggesting modifications to formal specification languages, which would make them more understandable. In particular, we have been suggesting alternatives to logical quantifiers. This will continue next year.

The third and fourth areas deal with the syntactic and semantic components of an experimental software tool. Its purpose is to test our solutions to a handful of problems in transforming English descriptions to formal specifications under significant user assistance. This work is in progress this year, but will not be complete until the end of another year's effort.

In dealing with syntactic ambiguity, one of the solutions that many have speculated about is the use of a semantic component to reject anomolous parses; we intend to test its effectiveness using the RUS grammar. When ambiguity is not resolved, questions must be presented to the user for his/her selection of the intended interpretation. Partial heuristics for this are part of the results of this year's effort. They will be implemented and tested during the next year. In addition, modification of the RUS parser and dictionary for the domain of software specification will continue during the next year.

Another problem is reflecting the conceptual view of a person in the semantic component. This must then be translated into the more mathematical view of present specification languages. On this problem, we have focused on designing this conceptual view; in the next year we will implement the translation from it to the mathematical level. A second semantic problem is anaphora resolution; the heuristics of Sidner (1979) are being modified for dealing with texts of specifications rather than dialogues (the class of discourse for which they were originally developed). This work will also continue next year.

4. References

Barstow, David. "A Knowledge-based System for Automatic Program Construction", Proceedings of IJCAI-77, 1977, 382-388.

Bobrow, R., "The RUS System", in Research in Natural Language Understanding, by B. Webber and R. Bobrow, BBN Report No. 3878, Bolt Beranek and Newman, Inc., Cambridge, MA, 1978.

Ford Aerospace, "Secure Minicomputer Operating System (KSOS): Computer Program Development Specifications (Type B-5)," Tech. Report No. WDL-TR7932, Ford Aerospace & Communications Corporation, Palo Alto, CA, 1978.

Horowitz, Ellis and Sartaj Sahni, Fundamentals of Data Structures, Computer Science Press, Inc., Woodland Hills, CA, 1976.

Johnson-Laird, P. N. "Mental Models in Cognitive Science", Cognitive Science, 4, No. 1, 1980, 71-115.

Joshi, Aravind K. and Scott Weinstein. "Control of Inference: Role of Some Aspects of Discourse Structure - Centering". Proceedings of the Seventh International Joint Conference on Artificial Intelligence, American Association for Artificial Intelligence, Mento Park, CA, 1981, 385-387.

Neumann, Peter G., Robert S. Boyer, Richard T. Feiertag, Karl N. Levitt, and Lawrence Robinson, "A Provably Secure Operating System: The System, Its Applications, and Proofs," SRI Project 4332, Final Report, Stanford Research Institute, Menlo Park, CA, 1977.

Parnas, D. L., "On the Design and Development of Program Families," IEEE Transactions on Software Engineering, SE-2, No. 1, March, 1976, 1-9.

Parnas, David L., "The Use of Precise Specifications in the Development of Software," Information Processing 77, B. Gilchrist, (ed.), North-Holland Publishing Company, New York, 1977.

Roubine, Olivier and Lawrence Robinson, "SPECIAL Reference Manual", Technical Report CSG-45, Stanford Research Institute, Menlo Park, CA, August, 1976.

Sidner, Candace Lee, "Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse," AI-TR 537, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1979.

Thomas, John C. "Quantifiers and Question-Asking," RC 5866 (#25388) IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1976.

Wason, P. C. and P. N. Johnson-Laird. Psychology of Reasoning: Structure and Content, Cambridge, MA: Harvard University Press, 1972.

Weischedel, Ralph M. "Prerequisites to Deriving Formal Specifications from Natural Language Requirements: Final Report", Dept. of Computer & Information Sciences, University of Delaware, Newark, DE, 1980.

Weischedel, Ralph M. "Reducing the Effort in Creating Formal Specifcations of Software Modules," Proceedings of the Conference on Information Sciences and Systems, Johns Hopkins University, Baltimore, MD, March 26-28, 1981a, 86-90.

Weischedel, Ralph M. "Practical Issues in having a usable Library of Software Specifications", Department of Computer & Information Sciences, University of Delaware, Newark, DE, 1981b.

Woods, William A. "Theoretical Studies in Natural Language Understanding: Annual Report, 1 May 1978 - 30 April 1979", BBN Report #4332, Bolt Beranek and Newman, Cambridge MA, 1979.

5. List of Presentations and Reports Generated this Year

Joshi, Aravind K. and Scott Weinstein. "Control of Inference: Role of Some Aspects of Discourse Structure - Centering". Proceedings of the Seventh International Joint Conference on Artificial Intelligence, American Association for Artificial Intelligence, Menlo Park, CA, 1981, 385-387.

Weischedel, Ralph M. "The Problem with Software Specifications & Two Emerging Solutions", Presented to the Computer Science Research Group, Ford Aerospace & Communications Corporation, Palo Alto, CA, August, 1980.

Weischedel, Ralph M. "Two Possible Uses", First KL-one Workshop, Jackson, NH, October 15-17, 1981. (Oral presentation only; no proceedings was published.)

Weischedel, Ralph M. "Reducing the Effort in Creating Formal Specifcations of Software Modules," Proceedings of the Conference on Information Sciences and Systems, Johns Hopkins University, Baltimore, MD, March 26-28, 1981, 86-90.

Weischedel, Ralph M. "Practical Issues in having a usable Library of Software Specifications", Department of Computer & Information Sciences, University of Delaware, Newark, DE 1981b.

Additionally, Ralph M. Weischedel was a participant in the Sublanguages Panel of the Applied Computational Linguistics in Perspective Workshop, June 26-27, 1981. The workshop was sponsored by the Office of Naval Research and the National Science Foundation. His experience in the style of English in software specifications was presented there. An overview of the workshop as a whole is to be submitted to the American Journal of Computational Linguistics by the workshop organizer Carroll Johnson.

DATE
FILMED

8